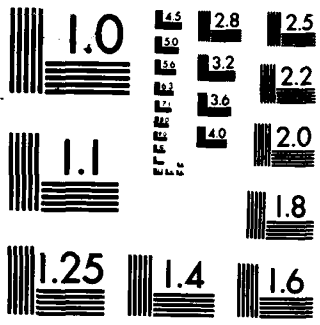
1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3

STARS TPP (8/06/86)

2

SOFTWARE TECHNOLOGY FOR ADAPTABLE, RELIABLE SYSTEMS (STARS)

TECHNICAL PROGRAM PLAN

6 AUGUST 1986

AD-A173 856

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION: UNCLASSIFIED	1b. RESTRICTIVE MARKINGS: NONE
2a. SECURITY CLASSIFICATION AUTHORITY: OASD-PA 86-3074	3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release, distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE: N/A	
4. PERFORMING ORGANIZATION REPORT NUMBER(S): STARS TPP - (8/06/86)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION: STARS JPO; 6b. OFFICE SYMBOL *(If applicable)*: SJPO	7a. NAME OF MONITORING ORGANIZATION: STARS JPO
6c. ADDRESS *(City, State, and ZIP Code)*: Office of Secretary of Defense, OUSDRE(R&AT/CET), The Pentagon, Rm. 3E114, Washington, DC 20301	7b. ADDRESS *(City, State, and ZIP Code)*: SAME
8a. NAME OF FUNDING/SPONSORING ORGANIZATION: STARS JPO; 8b. OFFICE SYMBOL *(If applicable)*: SJPO	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER: NONE
8c. ADDRESS *(City, State, and ZIP Code)*: SAME	10. SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT ACCESSION NO.
63756A	STARS	NONE	NONE

11. TITLE *(Include Security Classification)*
Software Technology for Adaptable, Reliable Systems (STARS) Technical Program Plan

12. PERSONAL AUTHOR(S)
Greene, J.S., Jr., T. Probert, W. Riddle, C. Giese, T.L. Quindry, J. Trimble

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT *(Year, Month, Day)*	15. PAGE COUNT
PLAN	FROM N/A TO	1986 Aug 6	40

16. SUPPLEMENTARY NOTATION
Ada is a Registered Trademark of the U.S. Government (Ada Joint Program Office)

17. COSATI CODES

FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
STARS, Software, Ada, Mission Critical, MCCR, Productivity, Reliability, Quality, Reusability, Standardization.

19. ABSTRACT *(Continue on reverse if necessary and identify by block number)*
Software Technology for Adaptable, Reliable Systems, or STARS, is the Defense Department's program to achieve dramatic improvements in software quality and to mitigate runaway software costs. This technical plan unifies and focuses the STARS program by emphasizing the use of Ada® and related software engineering technology and introducing the software first systems acquisition approach.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT
☐ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS

21. ABSTRACT SECURITY CLASSIFICATION: UNCLASSIFIED

22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE *(Include Area Code)*	22c. OFFICE SYMBOL
Col Joseph S. Greene, Jr.	(202) 694-0211	SJPO

DD Form 1473, JUN 86 *Previous editions are obsolete.* SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED

DD Form 1473 Instructions, JUN 86

EXECUTIVE SUMMARY

Software Technology for Adaptable, Reliable Systems, or STARS, is the Defense Department's program to achieve dramatic improvements in software quality and to mitigate runaway software costs. The Under Secretary of Defense for Research and Engineering (USDRE) directed the STARS Program Director to expeditiously develop and annually update technical and management plans for approval by the STARS Executive Committee. This technical plan responds to the USDRE direction.

The STARS Charter [1] signed by the USDRE on 1 November 1984, the Deputy Secretary of Defense program clarification memorandum [2], signed on 12 August 1985, and the requirements document prepared by the Services [3] provide the approved high level statements of STARS program requirements, objectives and approach. The technical guidance from references 1 and 2 is presented next in outline form to summarize the approved basis for this STARS technical plan.

- REQUIREMENTS
  - Future weapon systems require:
    - -- Adaptability and reliability.
    - -- New and enhanced capabilities.
    - -- Significant quantities of software.
  - Software cost projections require:
    - -- Unusual efforts to mitigate rising costs.
- OBJECTIVE
  - Long term savings and reliability benefits.
  - Major advances in the software engineering process.
  - Dramatic improvements in:
    - -- Software engineering productivity.
    - -- Software quality and reliability.
    - -- Time and cost of Defense software.
  - Transition of new processes into use.
- APPROACH
  - Apply sufficient resources in a concentrated R&D effort.
  - Exploit new software and systems opportunities for:

-- Commonality.
-- Standardization.
-- Portability.
-- Reusability.

- Use a focused management approach.
- Assign Service and Defense Agency organizations responsibility for technical direction and contract management.
- Develop major products under competitive industry contracts.

This technical plan unifies and focuses the STARS program by emphasizing the use of Ada® and related software engineering technology, introducing the software first systems acquisition approach, and establishing a strong industry leadership role in STARS technology development.

Specifically, STARS will:

- Demonstrate emerging STARS technology by sponsoring at least twelve shadow projects with Service Systems Program Offices at a rate of three per year. Shadow projects are implementatations of operational mission software using emerging STARS technology. These projects will be conducted in parallel on a non-interference basis with ongoing system development efforts by selected System Program Offices of the Services. Shadow projects will provide a pragmatic assessment of STARS progress and will help transition STARS technology into Mission Critical Computer Resources (MCCR) programs.

- Deliver several prototype automated software engineering environments in FY 1988 and fully operational environments in FY 1991 that run as distributed applications over networked heterogeneous hardware, exploit standard virtual interfaces, provide an object-oriented view to developers, provide configuration control of developed software, and support a design-by-successive-refinement software-first strategy.

- Develop and demonstrate a software-first technology with special emphasis on adaptability to reduce software costs and reliability to improve software quality.

®Ada is a registered trademark of the U.S. Government (Ada Joint Program Office)

- Develop and support an adaptable software technology through several initiatives:

  -- Develop a significant foundation of reusable Ada software in FY 1986 and FY 1987 for MCCR mission applications and software engineering support.
  -- Unify a consistent set of commercial functional interface standards in Ada as a basis for software component compatibility, adaptability and reuse.
  -- Provide a MILNET-accessible repository with appropriate access controls to support software reuse. Because STARS will seek to demonstrate and support software reusability opportunities to reduce mission applications software costs, the repository will include a significant quantity of mission applications software that can be used to evaluate and advance software development approaches for reusable software.

- Develop and demonstrate a reliable software technology through several initiatives:

  -- Develop and demonstrate computer aided tools to automate and unify software system definition, specification, design, coding, testing, maintenance and documentation, and to provide software life-cycle configuration control over all aspects of the software process. Validation capabilities will be included.
  -- Demonstrate the feasibility of extending the mechanisms used by Ada compiler builders in linking Ada package specifications and package bodies to provide comparable processes for assuring completeness and consistence between other phases of the life-cycle process.
  -- Contribute to the development of formal verification methods and supporting tools for MCCR systems developed in Ada.

- Pioneer and demonstrate a new way to treat high-risk software issues in MCCR acquisition.

STARS TPP - (8/06/86)

The STARS Executive Committee has approved this plan for execution of the FY 1986 program. Written line-in, line-out comments with rationale are invited and should be forwarded to the STARS Director, The Pentagon Room 3D139 (Fern C107), Washington, D.C. 20301 by 30 July 1986 for consideration in preparing the FY 1987 annual update.

Submitted:

[illegible]

Joseph S. Greene, Jr.
Colonel, USAF
Director, STARS

Approved:

[illegible]

Ronald L. Kerber
Chairman, STARS
Executive Committee

TABLE OF CONTENTS

SECTION 1 - BACKGROUND

This document is the top-level technical program plan for the STARS program. It describes the objectives of the program, the technical approach to achieve the objectives, the products to be delivered, and the product milestones and funding requirements. The requirements and needs which drive the program may be found in reference 3. Program management is described in a separate management plan [4].

1.1 SOFTWARE ENGINEERING CRITICALITY

DoD's weapon and warfare support systems depend on Mission Critical Computer Resources (MCCR) not just for their initial operational capability, but also to provide the capability to respond to changing operational threats and employment doctrine. Future national security will require new and enhanced high-technology weapons, command control and intelligence gathering systems that will depend on significant quantities of software. Without fundamental change to the software engineering process, DoD's annual expenditures for MCCR are projected [5] to increase from approximately $3 billion in 1980 to $11 billion in 1985 and then up to over $30 billion in 1990. Only a 20 percent increase in software productivity by 1990 would equate to cost saving sufficient to buy a new ship for the Navy, 15 new fighter aircraft for the Air Force and a tank battalion for the Army every year. We should be able to do considerably better than 20 percent just by using Ada software engineering and institutionalizing software reuse. Today, however, software development and maintainance ability is the critical-path activity limiting progress in many DoD (MCCR) programs, from affordability, risk management and capability points of view.

Not only will future systems require more software at a higher aggregate cost, but the software will be responsible for providing more and more of the functionality in weapon systems. Much of the growth in the power and sophistication of U.S. weapon and warfare support systems has been due to the extensive application of computers, particularly hardware technology. Software productivity increases have simply been inadequate to meet the growing complexity and size of military systems. As a result, software problems continue to grow more serious. Symptoms of these problems are seen in slippages in weapon system development schedules, operational system failures, weapon system inability to meet changing requirements, and soaring life cycle costs [3, 6, and 7]. The Software Technology for Adaptable, Reliable Systems program, called STARS, is the

DoD's program to achieve dramatic improvements in software quality and to mitigate runaway software costs.

1.2 SCOPE OF THE STARS PROGRAM

With planning initiated in CY 1981 and continuing through FY 1983, the STARS program experienced a slow and ambiguous start. In FY 1986, following major reviews [8, 9] STARS was redirected (See Section 3.3.7 below) and challenged to find and demonstrate ways to significantly reduce MCCR system software costs by CY 1991. New technical and management approaches were formulated to achieve major productivity enhancement. The new STARS technical approach will focus on use and extension of the Ada software engineering opportunities. The STARS management approach will exploit industry leadership in software research. Laboratories of the Military Departments will have a stronger role. STARS work will emphasize full and open competition continuing through the life of the program. STARS will foster the needed technology by funding research and development, integrating the contributions of many industry organizations and making those results conveniently available to the U.S. software industry. A research thrust is required because the ability to produce such a software engineering approach does not now exist. However, DoD software initiatives over the past fifteen years support the feasibility of the planned software-first thrust with high expectation of significant near-term results.

STARS cannot solve the DoD's software problems by itself. The STARS program must be built upon ongoing DoD programs and must leverage the American commercial and industrial base. Tight coupling exists between STARS and technology-based programs within the DoD and other Government Agencies. These include the introduction and maintenance of the Ada computer programming language by the Ada Joint Program Office, the efforts of the Joint Logistics Commanders (JLCs) to improve software acquisition practices, the establishment of the Software Engineering Institute (SEI), the establishment of life cycle software engineering support centers by the Services, and the Strategic Computing Initiative by the Defense Advanced Research Projects Agency.

1.3 STARS FUNDING

The funding profile for the STARS program is shown in Figure 1-1.

RDT&E	P.E. No.	FY87	FY88	FY89	FY90	FY91	FY92
Current Program	63756D	35.220	35.522	37.266	38.889	0	0

FIGURE 1-1: STARS Funding Profile for FY 1987-1992 ($ In Millions)

SECTION 2 - OBJECTIVE

The objective of the STARS Program is to achieve a dramatic improvement in our ability to provide and support software meeting mission critical defense requirements. This improvement will be reflected in the cost, schedule, and quality of MCCR software. The program will seek major improvements by the early 1990's.

2.1 DEFENSE SYSTEMS OF THE NINETIES

The STARS Technical Program Plan and implementation strategy have been developed based on a vision of the computing requirements for defense systems in the 1990's [6, 7]. By 1990 mission-critical defense systems will in general, exhibit the following characteristics:

(1) MCCR embedded in weapon systems will make routine use of multi-processors and networked or parallel architectures. Similarly, Command and Control systems will be based almost exclusively on distributed processing architectures.

(2) Data bases used by the commander will be distributed. Extensive use of computer networks rather than hierarchical structures will be used.

(3) The processing power of the computers used in defense systems will be increased by at least an order of magnitude. If the size and complexity of software grow at the same rate, the point will be reached beyond which continued implementation of software systems by past methods may become impractical.

(4) The amount of on-line computer-managed data will vastly increase. The automated systems supporting the commander must often sort through and reduce myriads of data to locate and present only the critical elements of information.

(5) The costs of the computer hardware used for software development and used in system acquisition and deployment will continue to decrease. The cost to develop and maintain software will be a principal factor affecting MCCR weapon system affordabilty and capability.

The STARS strategy contained in this plan responds to these critical factors affecting weapons system technology. The maturation of software technologies permitting the use of distributed computer networks, real time processing with multiprocessors, information protection for high integrity and trusted systems, and cost effective software generation concepts are topics of high interest.

2.2 CAPABILITIES NEEDED

The capabilities needed to answer the software challenges outlined above are aimed at positioning the DoD as an effective buyer of software, ensuring the availability of software support systems that are adaptable to each project, providing for the easy and trusted reuse of existing software, introducing a variety of rapid and cost effective software production and support techniques, and assessing and reducing the risks inherent in the development of new software systems. These capabilities include methods, techniques, and tools to:

(1) Provide for routine use of automation in the software creation and evolutionary support process.

(2) Reduce the cost of software development through the development and application of accessible, trusted, reusable software components.

(3) Improve the software creation and evolutionary support processes with software-first design, specification and documentation methods.

(4) Reduce the risks inherent in the development of mission critical software through the use of appropriate software standards and procedures.

(5) Demonstrate the transition of new technology to practice on DoD MCCR systems and use the feedback from lessons learned to improve the technology.

SECTION 3 - TECHNICAL STRATEGY

Severe software problems are not unique to a single Service or Defense Agency. In fact, similar software problems have been experienced by every Defense Department component. These common problems deserve a common solution. The STARS program will concentrate resources in order to develop and make available solutions to support the entire DoD.

The STARS program will not supplant existing programs in the area of software engineering. Rather, STARS will supplement other efforts with a focused and compatible program. Related Service and Agency software programs must continue as they directly complement STARS in the near term and, in the long term, will provide the follow-on and continuity. The thrust of STARS is to develop and productize software solutions to be used by commercial industry. STARS will be successful when the software technology base supporting DoD contractors and the DoD has been raised and the process of maintaining and further improving that technology base has been commercialized by industry.

3.1 LEVERAGE APPROACH

The DoD requires quality products to support efforts to develop and maintain operational MCCR systems. "Quality products" means that the tools and procedures are not only feasible and applicable, but acceptable for use (i.e., cost-effective, adaptable, reliable, maintainable, and reusable). Sometimes the term "production quality" is used to describe these attributes.

There is an important distinction between promoting technology and commercialization of software support systems that meet the varied and specific needs of the DoD. The STARS approach addresses several facets of the software problem: basic technology, engineering the technology into products, and technnology insertion. The people and the organizations who develop and maintain DoD systems have also been considered because products alone fall short as a total solution. Technology transition and insertion do not occur naturally. Although Defense MCCR systems are built by the U.S. weapon systems industry, the defense market alone is not sufficient to stimulate the needed change. We must foster change that creates competitive advantage for industry in the commercial sector as well. For this reason, the STARS program has been recently focused to provide U.S. industry a leadership role in improving the defense software development process. The STARS strategy includes explicit linkages to commercial opportunities through emphasis on commercial as well as government software standards.

3.2 PRODUCTIVITY-IMPROVING OPPORTUNITIES

STARS will develop and demonstrate ways to significantly increase software productivity over the next 5 years. The DoD will need major increases in software productivity, relative to today's baseline, to get affordability and capability improvements at the same time. A high, near-term productivity objective also serves as a convenient criteria against which to evaluate the potential contributions of candidate projects and with which to focus the STARS program. STARS will seek major and early productivity achievements by exploiting opportunities from several different sources. These are:

- Ada Features. The Ada language itself provides powerful features that contribute to new levels of productivity in the design, programming and maintenance phases. Although the language is only now beginning to get wide use, reviews [10, 11] of early Ada products provide encouraging evidence that Ada will achieve the productivity improvement sought by the Ada language designers. These results have been achieved generally by people without prior Ada experience using early compilers on conventional machines, so even greater improvement factors are reasonable to expect.

- System Size. Software engineers have long recognized the principle that the best way to reduce software cost in all phases of the life-cycle is to reduce the size of the software providing the same capabilities. The Ada language designers had this principle in mind, particularly for systems in the large. Because few very large systems have been developed in Ada, the opportunities provided by Ada to reduce system size have not yet been widely recognized. Much of the early Ada code developed by industry and government and delivered to the Ada repository on SIMTEL 20 was a direct redesign and implementation of capabilities already existing in other languages. Comparison of the new Ada code to previous versions in other languages has provided evidence for expecting smaller Ada programs as compared to the same function in another language [12].

- Reuse. Software engineers have also recognized that reusing, rather than rebuilding, would be a powerful software approach with significant cost-benefit opportunities [14]. Unfortunately, the common past practice of using many different languages and language versions, as well as efforts to increase performance through use of machine and operating system dependent features, has generally discouraged widespread reuse. Given the past software base, reinvention has generally been cheaper than reuse. The Ada language with a consistent application of design and coding practices that improve readability, isolate machine and operating system dependencies, and encourage Ada source-level portability should provide a major new opportunity for mission applications software reuse. STARS will seek to develop a sizable amount of software in ways that will encourage and foster reuse.

Reusability oportunities will be exploited by STARS at several levels. As a first step, having a large quantity of software covering many different application domains in a common language, with clearly isolated and well documented treatment of machine and operating system dependencies would provide a totally new opportunity to exploit reusability. As a second step, unification of functional interface standards would introduce additional opportunities for reuse. For example, subsystem components could be more easily separated for reuse. Also, a new and more specific taxonomy of software could be developed to facilitate description, cataloging and retrieval of software components. Automated tools to reconstruct (to the extent possible) requirements-level, functional-level and specification-level descriptions by processing source, object, and command language code would further facilitate reuse. At another level, approaches that would reuse trusted subsystems components, their designs and specifications, and their formal proofs needs to be explored. There are then several different opportunities for reuse by the software engineer at subsystem, component, package, and subpackage granularities. STARS will exploit reuse at all levels that provide cost-benefit returns.

- Labor Saving Tools. The software community has recognized the opportunity to increase productivity by capitalizing the programming force with tools and special architecture software development machines. The programmer support capability possible for a given cost has increased steadily over the years and is predicted to continue to increase [13].

In the labor saving area of opportunity, repository technology to facilitate sharing, transfer and control of software during development, maintenance and reuse will also be developed.

Efforts to reduce the overhead of documentation and project management will also be undertaken, but are not expected to provide comparable productivity contributions. Much more powerful opportunities seem to lie in the direction of changing the software engineering process so that these former, human-intensive, paper driven controls are no longer required.

Concurrent advance along these several dimensions will combine to give a compound productivity improvement of at least a factor of 10.

3.3 SOFTWARE-FIRST PROCESSES

Traditional system acquisition processes within the Department of Defense (DoD) have tended to emphasize hardware over software. Hardware has always been part of systems acquisition whereas software has only recently gained a position of prominence. As a result, acquisition processes have been developed, and evolved, with hardware primarily in mind. Software has too often been viewed as an "add-on" to be acquired after the hardware is specified or obtained.

The acquisition process has been further complicated by a common practice of considering hardware to include a run-time support system upon which application-dependent software must execute. In keeping with the hardware-first nature of traditional acquisition processes, the run-time support system is usually specified, and sometimes fully acquired, before the application-related software issues are addressed.

High levels of software cost reduction are believed possible by addressing software, rather than hardware, first in the acquisition of an automated and MCCR systems. Considering software first can lead to a much richer set of possibilities for software reuse. Software first can provide more extensive and easier determination of user requirements before the constraints imposed by the hardware complicate the problem. Considering software first will also allow the early determination of the software's general structure and will therefore help specify hardware facilities (such as distribution, parallelism, etc.) needed to achieve desired performance levels.

The STARS Program will promote processes that support a software-first approach to systems acquisition. The processes are characterized by the following attributes that:

- Provide for early and frequent reassessment of user requirements.
- Provide a "bi-modal" process by which a system can be developed through the free intermixture of bottom-up and top-down processes. The bottom-up process will provide supporting tools to compose new systems from reusable software and to reconstruct (as much as possible) specification level, and requirements-level descriptions from the code itself. The top-down processes will proceed in the classical sense from a requirements view through specification to code itself.
- Allows and encourages the use of prototyping for requirements definition, decision exploration, and system evolution.
- Incorporate both formal and computer aided analyses.

Processes with these characteristics are fundamentally different from current management-oriented software processes that lead to the development of elaborate specifications before the fundamental problems concerning a software system are identified and understood.

Software-first processes require that the implementation language be machine-independent. This, in turn, requires a strict and enforceable control of the language definitions and implementation. The DoD common high order language, Ada, defined by the Ada language standard [15] and controlled through the DoD trademark, "Ada", and the compiler validation facility, is the most uniformly implemented language ever produced. Ada provides the best opportunity to support a software-first technology today.

The STARS Program will emphasize use of an adaptable (reusable) software technology process to replace the present life-cycle software model. Initial efforts will be compared to the life-cycle phases defined by traditional "waterfall" models such as MIL-STD-2167 [16]. Attention will, however, be given to an Ada-focused alternative model supporting an evolutionary development process. The STARS approach will make maximum use of Ada for specification, design, implementation, execution, and evaluation of MCCR software. Support tools will also be implemented in and for Ada.

3.4 Ada - THE LANGUAGE OF CHOICE

Ada provides an integrated collection of powerful, advanced features (typing, encapsulation, generics, exception handling, tasking, packages, specifications, etc.) selected for the development and maintenance of the software typically found in MCCR systems. Ada provides a well controlled, standard implementation that permits high degree of source-level portability between validated compilers and provides opportunities to isolated machine and operating system dependencies to achieve a high degree of machine independence.

The Ada language brings with it a number of technologies that reduce the cost of implementing software. Beyond the clear impact afforded by software reuse, Ada supports a number of design techniques that reduce the risks of software development.

The goals of the STARS Program -- increased productivity and increased MCCR software quality -- were fundamental to the design goals of the Ada language. Thus it is natural that the STARS Program capitalize on the base provided by Ada activities. Ada can be used to capture the results of the STARS Program and share them throughout the DoD software community. Ada will provide a common medium through which many different avenues for software technology advancement can be consolidated and focused on the problem of increasing the productivity of the MCCR software production process and the quality of the resulting MCCR software.

3.5 STANDARDIZATION SUPPORT

The problems of software portability and communication within and across computing architectures have existed in the information processing community from the beginning. Industry and government have developed a comprehensive suite of over 1,000 standards that address software related issues [17]. Employment of technically and functionally appropriate standards has several benefits. First, standards allow the establishment of consistent functional interfaces that support well-layered design and development of components. Second, standards facilitate the reuse of properly designed software components across application domains. Finally, standards capture the fruits of an extensive effort already undertaken by industry and government, thus avoiding replication and re-invention.

STARS will therefore foster the use and development of government (e.g., Ada, CAIS, DIANA, etc.) and commercial information processing standards as an important part of the STARS Program. The goal of the STARS initiative will be to unify a consistent set of standards through bindings and implementations in and for Ada. The fact that STARS will encourage company sponsorship of STARS funded efforts before accredited commercial standards bodies is a key aspect of the STARS leverage strategy for in stimulating commercialization of STARS initiatives and fostering a new private sector markets for the software industry.

3.6 RELATIONSHIPS TO CERTAIN OTHER PROGRAMS

Since the initial definition of the STARS Program in 1983, several other programs in software have started within and outside the Government. Collectively, they provide a context for consolidating STARS program efforts. Three activities are of particular importance in this regard. The Software Engineering Institute (SEI) has the role of transitioning advanced software technology into widespread practice within the DoD software community. The Strategic Computing Program (SCP), within the Defense Advanced Research Projects Agency (DARPA), has developed plans for rapidly capitalizing on advanced software technology at the edge of the current technology horizon. Several programs such as the Ada Joint Program Office (AJPO), World Wide Military Command and Control System modernization (WIS) and the Strategic Defense Initiative (SDI) seek to capitalize upon more well-developed technology to achieve immediate gains.

The STARS Program will, in general, coordinate and integrate STARS activities with those of other programs. The coordination and integration with respect to the Service programs and SCP, in particular, are indicated in Figure 3.1. The Figure illustrates the point that STARS program activities will fill the gap between the immediate results achieved within the Service programs and the long-term results hoped for through the SCP program. The bulk of STARS activities will be oriented towards medium-term results that will deliver a significant productivity increment.

Coordination and integration with SEI will result in the de-emphasis of transition activities within the STARS program. The initial plans for STARS assume that the task of transitioning new technology developed by STARS will be handled by the SEI. The STARS Program, however, will retain activities called shadow demonstrations to assess the value of STARS technology in real-world situations. A secondary intent of shadow demonstrations will be to provide guidance for STARS program planning and execution. The data collected through demonstration will provide a pragmatic measure of the Program's progress and also help identify high-payoff directions from among alternative activities.

3.7 RESTRUCTURING OF THE FY 1986 PROGRAM

The program concept is consistent with the long-standing goals of the STARS Program, but represents a significant restructuring and focusing of the technology and management approach. Prior to the restructuring of the FY 1986 program, STARS documents described the program in terms of eight different technology areas [18]. The previous Applications Specific and Measurements areas will be combined and replaced with a new emphasis on shadowing selected, real, mission-critical systems developments to provide STARS technology demonstrations. Shadow projects will be STARS-funded initiatives to develop operational-quality MCCR products by applying the Ada-based STARS development processes to real, DoD mission critical systems. The former STARS program technology areas called Systems, Methodology, Automated Environments and Man-machine Interface will be replaced by a single, focused, software-first technology development thrust. The former Business Practices and Human Resources areas will not be funded by STARS and will be referred to the Services and the SEI as technology transition activities.

Prior STARS efforts to automate the current state-of-the-art as represented by the Software Engineering Environment (SEE) efforts will be referred to the Services as commendable efforts, but inappropriate for STARS funding because they represent a technology approach insufficient to achieve STARS productivity goals. Service cost estimates of the STARS-SEE preliminary design also make the SEE unaffordable within the STARS program as previously planned. These conclusions do not mean the STARS will not develop software engineering support environments. As discussed in Sections 4.3.1 and 4.3.2, STARS will develop environments, but they will differ in two important ways from previously planned efforts. First, the new STARS environments will support a new software engineering approach. Second they will be developed themselves using that approach. This plan assumes that the change will allow the development of several environments for less cost then previously estimated.

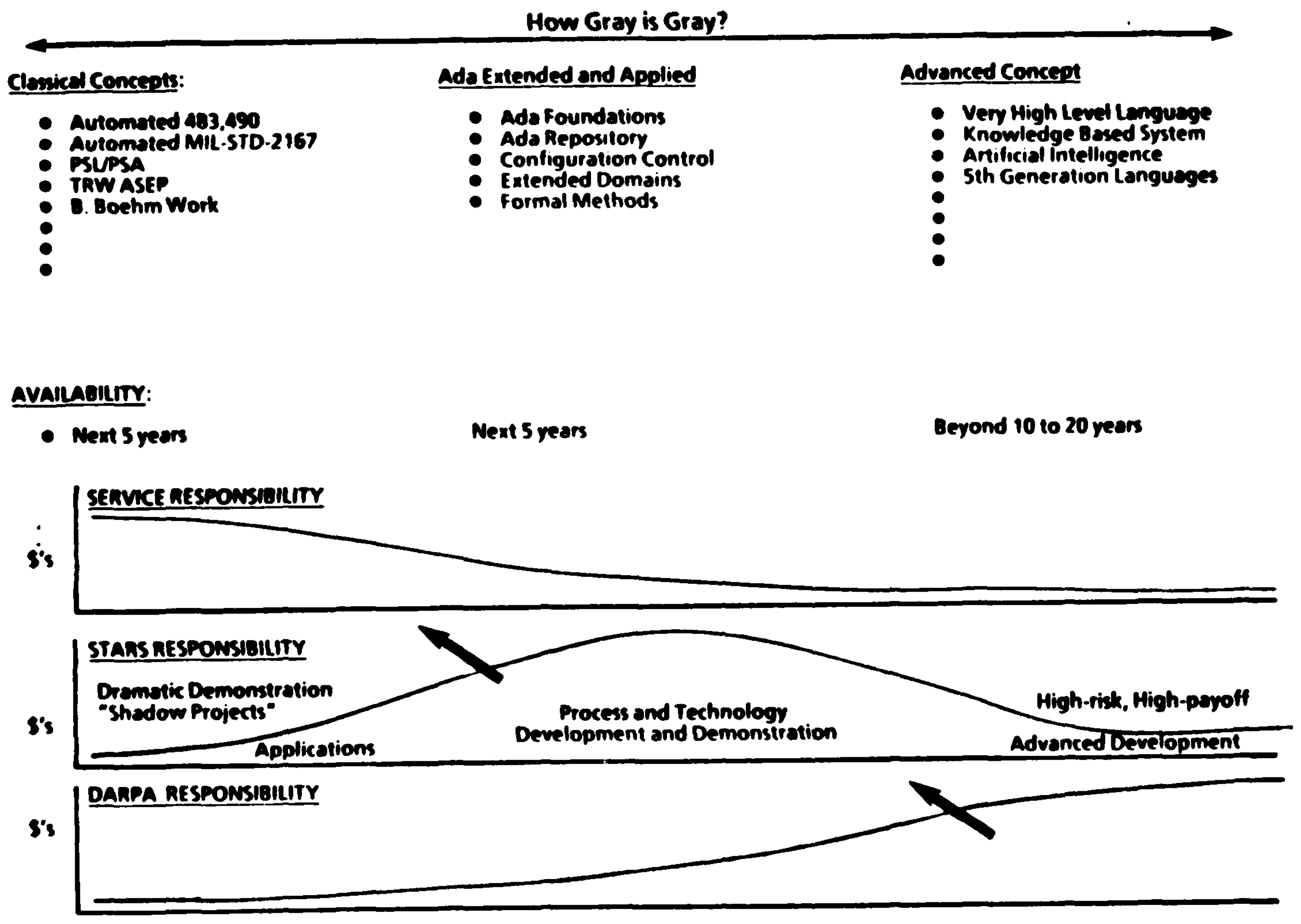

FIGURE 3.1 STARS PROGRAM FOCUS

Research in very high level languages, knowledge-based systems, artificial intelligence, 5th generation languages, non-von Neumann architectures, logic programming and symbolic evaluation will not be funded by STARS. Proposals for work in these areas will be referred to DARPA and the Services' 6.1 and 6.2 research programs. These advanced concepts, while potentially very promising, are not judged likely to meet STARS' delivery schedule.

SECTION 4 - TECHNICAL PROGRAM

The STARS program is a software engineering technology development program. The program builds on the 15-year foundation of a Defense Department initiatives that began in the early 1970's following publication of a major Air Force study [19] that identified software costs as a major problem for mission-critical systems. In particular, STARS will build on the Ada computer language and software engineering process.

4.1 PROGRAM STRUCTURE

To capture the opportunities of the Department of Defense Ada software engineering initiative, the STARS program is structured with four fundamental and related thrusts: (1) The success of the STARS-developed software engineering process will be measured pragmatically through "shadow" projects that develop real operational mission critical applications in Ada; (2) Software engineering environments will be developed that support a software-first approach to system acquisition; (3) The technology development thrust will focus on adaptable and reliable software engineering approaches; and (4) To solve fundamental research issues, STARS will pioneer an approach to risk reduction applicable to software-intensive mission critical systems. A software repository will be maintained to facilitate sharing of reusable software.

Each of these thrusts will be described in greater detail in the following subsections. The STARS Program is based on industry leadership through a few lead contracts and includes participation of Service Laboratories in research areas and Service Product Divisions in shadow projects.

The program resource allocation to these areas is shown in Figure 4-1.

STARS TPP - (8/06/86)

Purpose	FY86	FY87	FY88	FY89	FY90
Gramm-Rudman	2.0	-	-	-	-
FY85 Continuations	8.1	-	-	-	-
R&D by the Services					
Management	1.5	1.5	1.5	1.5	1.5
R&D	6.0	6.0	6.0	6.0	6.0
R&D by Industry					
Shadows	3.0	3.0	6.0	6.0	6.0
Foundations	10.0	3.0	-	-	-
Development	5.4	13.1	12.4	14.2	15.8
Risk Reduction	3.0	3.0	6.0	6.0	6.0
Repository	-	3.0	1.0	1.0	1.0
STARS JPO	2.5	2.6	2.6	2.6	2.6
Total	41.5	35.2	35.5	37.5	38.9

Figure 4-1 Current STARS Program Allocation ($s in Millions)

4.2 SHADOW DEMONSTRATIONS

Shadow demonstrations, using the discipline and processes developed by STARS, will be selected from weapons, command and control, and intelligence areas to provide representative demonstrations for each Service. Shadow projects will provide a pragmatic measure of the STARS program progress in developing and introducing a new software engineering approach, provide realistic and useful feedback to the technology developers, and contribute to the process of technology insertion. The "should-cost" goals for shadow developments will be set by the STARS Director and reduced over the program years to eventually demonstrate the productivity improvement sought. Shadow activities serve to demonstrate the feasibility and value of using STARS-developed software technology on real defense systems software projects. The tools and techniques underlying these demonstrations will be developed as part of the STARS technology development activities. The focus is on the actual demonstrations themselves.

Most candidates for shadow activities will be selected from system programs nominated by the Services and DoD agencies. The actual programs to be shadowed will be selected in coordination with the STARS Director, the Service or Agency, the System Project Office Director, and industry contractor involved. STARS will fund the shadow projects. The following criteria are a few of those that will be used to select STARS Shadow demonstration projects. The program *should:*

- Be in the weapons, command and control or intelligence area.
- Provide potential for software reuse.
- Use a traditional software acquisition approach with a non-Ada implementation.
- Have a projected size of 100K lines of code or more for the non-Ada implementation.
- Permit portions of the Ada shadow applications software to be placed in a STARS repository under appropriate access controls.

Reusable end-application software system components will be developed in support of these demonstration activities. The development of these components will therefore be driven and their definition will stem from actual projects. Coordination by mission needs among the various shadowing projects will result in the identification of generic components usable over a variety of projects and application areas.

4.2.1 1985-PLANNED APPLICATIONS

Several application activities initiated or planned in the FY 1985 STARS program result in the development and delivery of MCCR software in Ada and have therefore been selected for continuation. These are:

Common Ada Missile Packages (CAMP) will apply reusable Ada software concepts to DoD Missiles. CAMP will also develop and demonstrate a prototype DoD software composition system.

Ada Based Signal Processing is a software conversion demonstration of a data-flow graphic form technique for building adaptable systems to include Ada application areas for signal processing and C^3I.

Ada Based Integrated Control System (ABICS) enables the development and application of Ada software to flight-critical avionics systems. This project will demonstrate the application of reusable software parts to the Advanced Tactical Fighter.

Automation of User Requirements will investigate and develop an evolutionary software development capability as an alternative to the traditional, work product oriented approach. It will develop a technique for specifying and validating user requirements by which resulting designs can be easily (and possibly semi-automatically) translated into executable Ada code.

Convert-to-Ada Tools will develop and demonstrate a means for reducing the number of languages used in the DoD software inventory of Mission-Critical systems through automated conversion to the Ada language. Guides and tools will be developed to support decisions to convert or rewrite, and to provide automated aids for conversion.

The Computer Aided Instruction (CAI) project will conduct advanced experiments into the use of CAI as a reinforcement to platform instruction in Ada.

The Distributed Computer Development task will develop concepts for advanced Ada program development and support environments using new hardware and software technology.

4.2.2 FY 1986 SHADOW PROJECTS

To expedite the FY 1986 execution of shadow projects, STARS will fund Pilot Ada Capabilities Transition (PACT) Projects that have been formulated by the Software Engineering Institute (SEI). Candidate projects have been identified from each Service. The SEI will provide the infrastructure to oversee the FY 1986 PACT projects using commercially available Ada environments and state-of-the-practice software engineering techniques. This initial work will help establish a baseline for measuring future STARS progress and will provide feedback and lessons learned to help evolve the Ada software engineering process.

Pilot projects for shadowing will be established at several industry sites to produce mission critical software subsystem products in Ada. The SEI and STARS Joint Program Office will establish guidelines for the pilot projects. Three to six pilot projects will be selected, depending on the level of effort required. Coordination of the shadow pilot project efforts and technical support for the projects would be provided at the SEI. SEI people would be available to assist at the industry sites and will provide feedback to the SEI, AJPO, and STARS. Industry members may also work at the SEI as necessary.

Figure 4-2 lists the candidate PACT/Shadow projects that are being considered by the SEI:

Service	Project	Company
Army	FATDS	Magnavox
	CSS	To Be Determined
	PLRS	Hughes
Navy	BSY-1	IBM
	ISCS	Rockwell
	ACDS	Hughes
Air Force	Simulators	Boeing
	CSSR	GTE
	ATF	Grumman
	MILSTAR	Lockheed

Figure 4-2 Candidate FY 1986 Shadow Projects

4.2.3 SHADOW CRITERIA

Criteria for the selection of FY 1986 STARS-sponsored PACT/Shadow projects to achieve the STARS program objectives will be developed by the STARS Joint Program Office and coordinated with the Software Engineering Institute. Management of STARS Shadow projects after FY 1986 will be reviewed and updated to include the STARS lead contractors.

4.3 PRODUCT DEVELOPMENT

Powerful market forces are at play. The sweeping introduction of the personnal computer, the mass production of low-cost and powerful commodity software and the emergence of third-party software marketing organizations that have changed the computer business in ways to which no organization is immune. We cannot predict the timing, the basis, or the extent of similar forces in the future. We can predict that such phenomena will occur.

In picking a market place strategy for a program like STARS there are several dangers. Bucking the tide can be very costly. At the same time, locking in on yesterday's force can miss the next big opportunity. For example, were STARS to lock on to today's powerful commodity software market force as the key to tomorrow's success, STARS could miss the next fundamental new market force. If, for example, the industry provided the ability to adapt software to the unique requirements of each using organization in near real-time to give that organization a competitive edge, a new market with a service orientation could be added to today's commodity oriented market. Adaptability, in fact, is the capability that MCCR mission critical DoD systems need; and, therefore, is the market place change that STARS seeks to foster.

The STARS market strategy will include the development of many new products. The products that STARS will develop, while intended to be usable and useful in their own right, are intended more fundamently to be adaptable and reusable. The STARS product is not the end, but rather the means by which a new process of software engineering will be developed, demonstrated and explained. The STARS process will open up new markets and new opportunities for commercialization. That commercialization may be a very different service oriented opportunity that goes beyond the capabilities that are possible solely in a product oriented commodity market.

STARS product development will be done in a way that seeks to take the early cost risk out of a new process, but allows maximum opportunity for industry to extend the early efforts. Widespread sharing of the results of work funded by STARS is essential as the catalyst by which a new adaptable technology process will be developed, understood and extended by industry.

The STARS process-oriented and service-based marketplace strategy underlies the choice of principles (Section 4.3.2.1) and technical guidance (Section 5). The principles and guidance as presented are not final, but should provide a strong signal to the technical reader that something different is going on in STARS. STARS seeks to provide cost leverage in mission applications software. The approach should also provide cost leverage in development of the support environments. The program includes the use and demonstration of the new processes in both domains.

4.3.1 TECHNOLOGY INTEGRATION

The STARS lead contractors [see reference 4] will each be responsible for integrating the results of research and technology development efforts (described in Sections 4.4 and 4.5 below) to produce software engineering support environments specialized for a particular application domain. Each environment should make maximum reuse of common Ada foundation capabilities (described in Sections 4.4.1.1 below). The responsibility to deliver these software engineering environments is intended to provide the lead contractors a practical focus for their research and technology oversight responsibilities. The software engineering environments are intended to be a demonstration of the software-first software engineering approach they seek to support.

4.3.2 MCCR SOFTWARE ENGINEERING ENVIRONMENTS

By the end of FY 1988, three prototype software engineering environments will be completed for peer review and evaluation. These systems will be built using common, reusable Ada software unified by a consistent set of functional interface standards to provide an adaptable framework for interfacing tools and processes. A configuration control discipline will include tools for the protection of software at different levels of configuration control and for promotion between levels of configuration control. The system will include documentation and specification approaches supported by computer aided processes. A significant and growing set of tools for reliability and adaptability processes will be defined and under development.

By FY 1991, three production quality software engineering environments will be delivered. These will be adapted to the unique requirements of specific application domains. The design and integration approaches used in the development of these environments should demonstrate a new level of adaptability based on a common, reusable software technology. The time and cost to develop future environments adapted to the special needs of a particular application area should be demonstratably reduced over today's practice. These <u>products</u> are in fact intended to be the <u>processes</u> by which future applications systems will be developed and maintained. The adaptable, reliable software-first technology assures that the process is more important than the application product at any one time because the process provides the capability to respond on useful time scales to changing threats.

4.3.2.1 FRAMEWORKS

A fundamental aspect of a software engineering environment centers around the framework that such an environment provides for supporting and interfacing the tools and capabilities within the engineering process. A number of principles will be adopted as technical invariants to guide the software engineering environment developments. As invariant principles, the development environments must:

- Use unmodified commercial operating systems.
- Use Ada as a command language through a common Command Language Interpreter.
- Use the underlying file system.
- Provide tools for protection of and promotion between configuration control levels.
- Use a validated Ada compiler.
- Use common command words consistent with Ada in new tools.

- Use the International Standard Generalized Mark-up Language for all document preparation.
- Provide a taxonomy and system to search for and access reusable software.
- Support the relocation of information between geographically distributed systems.
- Support the registration of abstract document types that would be available over data communications networks from a repository coupled with a generic editing capability specialized by the registered type.
- Be machine independent down to a minimal set of well defined, low level interfaces, where implementing bodies will respond to the specific environment.
- Interface to external facilities, when used, through standard interfaces for virtual terminal, network, data base, graphics and file naming conventions.
- Support outside data paths to allow relocation of information between distributed subsystems.
- Demonstrate access to a remote data dictionary that could be part of the repository system.
- Assure that management tools impose no additional burden on the programmer or slow the development process.
- Keep the environment overhead to not more than a factor of 2 over doing a like function with a manufacturer's development environment.
- Be hostable on any system sufficient to host a production Ada compiler.
- Allow hosting multiple environments on the same hardware without compromising information or reporting to a central control.
- Use an abstract Ada data dictionary for integration.
- Provide life-cycle support of software and its associated documentation including multi-level trusted configuration control of software and document versions.
- Be developed in an evolutionary manner, supporting extensibility of capabilities throughout the environments life.
- Use existing, off-the-shelf software whenever possible, both within the environment itself and in applications being developed under the control of the environment.
- Support the reuse of existing software by introduction of a reuse taxonomy and process extending from requirements to code and from code back to requirements.

4.3.2.2 CONTROL PROCESSES

An automated control process capability with clear levels of control (such as development, test and operation), and with a multi-level trusted promotion process will provide a fundamental contribution to software reliability. The characteristics needed will be investigated early in the STARS program.

4.4 TECHNOLOGY DEVELOPMENT

The Shadow activities discussed in the previous sub-section provide direct support for moving modern software technology into use throughout the DoD software community. Some of this technology already exists. Improvements can be obtained by developing and making available technology that is currently within the state of the art. Other improvements require investigations into fundamental issues. The STARS Program technology development, productization and research activities will provide future capabilities to be transferred into practice. This section discusses STARS technology development efforts that focus on adaptable and reliable software technologies.

4.4.1 ADAPTABLE SOFTWARE TECHNOLOGY

The STARS program will seek an ADAPTABLE SOFTWARE TECHNOLOGY by developing a critical mass of foundation capabilities in machine-independent Ada as the core of a reusable technology. Integrability and reusability will be enhanced through a strong focus on commercial standards (e.g., those of IEEE, ANSI, and ISO standards bodies) at the functional interface levels (as compared to the tools-to-environment level that has been the prior fixation of the STARS Program). A major thrust will be undertaken to provide a trusted repository to describe, locate and access end-application and tool software (within the constraints of export controls and security concerns). An Ada-based, bi-modal technology for design-by-successive-refinement will be developed to support software reusability concepts. Along with the process there will be an emphasis on elimination of the overhead burden of duplicative and often redundant documentation requirements.

4.4.1.1 FOUNDATION Ada CAPABILITIES

A prerequisite to an adaptable software engineering process is a critical mass of software that is available, organized and structured with a view to adaptability. Because Ada is relatively new and because widespread system development in Ada is really just beginning, the opportunity exists to achieve considerable economies by focusing on the common software found in almost every system. STARS will seek to develop such common software in a way that facilitates and encourages reuse. This common software, called foundation capabilities, includes work in areas such as:

- Command Languages
- Text Processing
- Graphics Protocols
- Operating System Capabilities
- Data Base Tools
- Network Protocols
- Planning and Optimization Aids

- Design and Analysis Tools

STARS will build on the efforts of many government contractors whose Ada products have been delivered over the past four years to the Ada repository available on SIMTEL20 through MILNET. The following list is representative of some of the products that have already been delivered or are under contract.

- Symbolic Debugger
- Automatic Path Analyzer
- Metric Instrumentation
- Source Formatter
- Path Analyzer
- Statement Profile Report
- Complexity Measures Report
- Cross Reference Package
- Compilation Order Report
- PDL Processor
- Design Requirements Traceability
- Documentation Manager
- Automatic Specification Analysis
- Complexity Measurement Analysis
- Graphics PDL Support
- Standards Checker
- General Management (Costing)
- General Management (Work Flow Control)
- Data Dictionary System
- Program Design Assistant
- Standards Support
- Documentation Reports
- Set, Package, and Subprogram Use Reports

Appendix A provides a brief summary of some of the foundation capabilities that are candidates for development with FY 1986 funding. The foundation work can be undertaken concurrently with other STARS efforts. For this reason, the work will be competitively contracted prior to the selection and award of the STARS lead contracts. These foundation capabilities will be subsequently modified (adapted) in FY 1988 to comply with function interface standards that are discussed in the next section.

4.4.1.2 STANDARDS

Over 1000 standards have been identified [17] that relate to software processes. STARS will seek to unify a meaningful subset of these standards through Ada implementation and bindings. The following list illustrates the kinds of standards that will be treated.

Programming

- Ada Programming Language
- Ada as a Common Command Language

- Ada as a Common Program Design Language
- Descriptive Intermediate Attribute Notation for Ada (DIANA)

Networks

- Internet (IP)
- Transmission Control Protocol (TCP)
- User Datagram Protocol (UDP)
- Teletype Network Protocol (TELNET)
- Internet Control Message Protocol (ICMP)

Applications

- File Transfer Protocol (FTP)
- Trivial File Transfer Protocol (TFTP)
- Simple Mail Transfer Protocol (SMTP)
- Virtual Terminal Protocol (VTP)
- Virtual Device Metafile (VDM)
- Document Content and Interchange Protocols
- Virtual Device Protocol (VDP)
- Remote Procedure Protocol

Text Processing

- Computer Language for the Interchange and Processing of Text (CLIPT)
- International Standard Generalized Mark-up Language

Graphics

- Graphics Kernel Standard (GKS)
- Ada Binding to GKS
- North American Presentation Level Protocol Standard (NAPLPS)
- Programmers Hierarchial Interchange Graphics Standard

Database

- Common Ada Programming Support Environment (APSE) Interface Set (CAIS)
- Initial Graphics Exchange System (IGES)
- Distributed Database Protocols
- File Server and Receiver Protocols

Figure 4.3 provides an example of the way in which commercial interface standards can be used to build generic system components. Once these concepts are used in practice for much of a system, for example, they can be developed to use graphics capabilities in software design which could be directly transitioned to a battle management graphics application. STARS will use commercial standards to foster such leverage through reuse across functionally diverse applications.

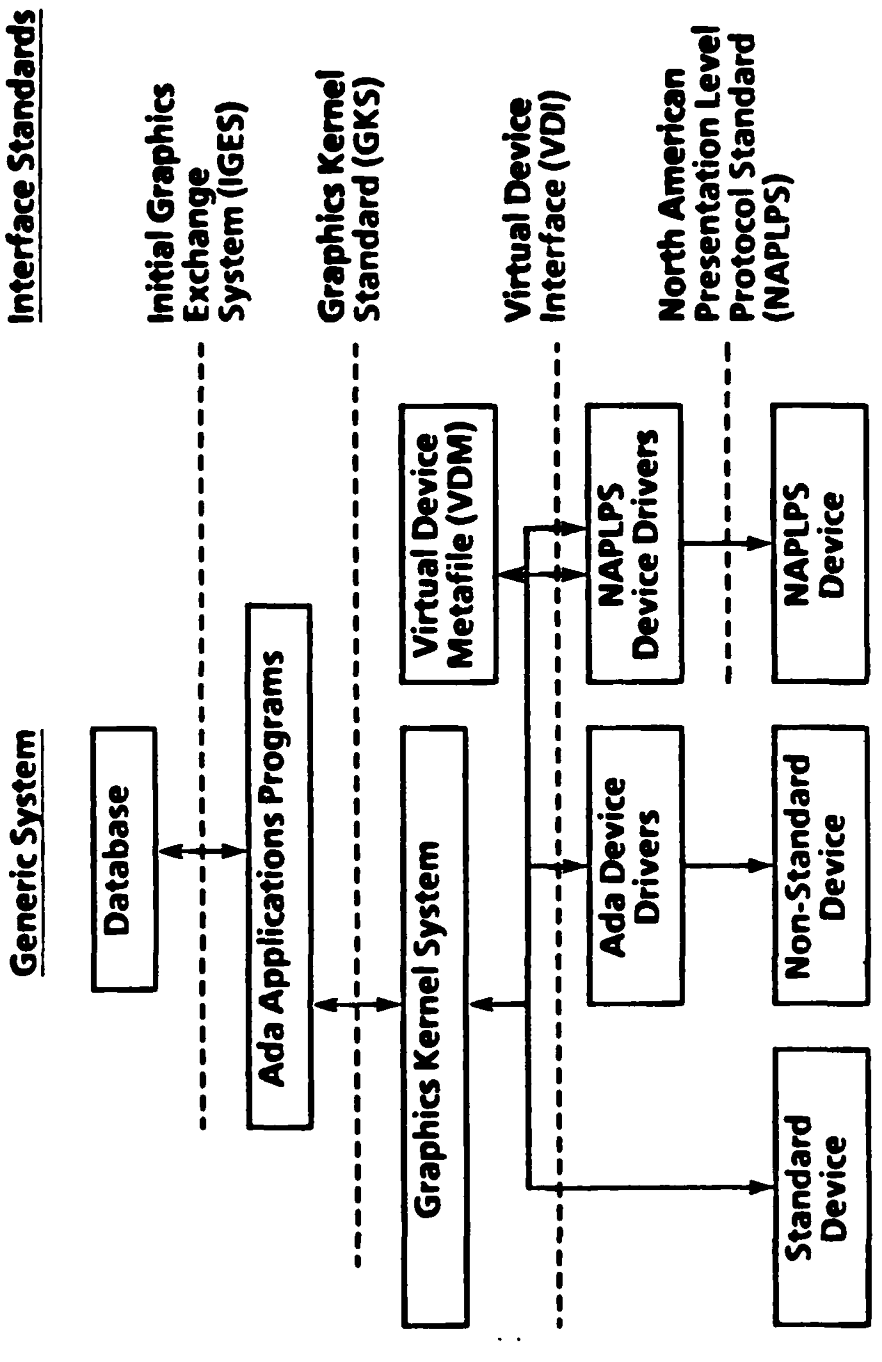
Generic System
Interface Standards
Database
Ada Applications Programs
Initial Graphics Exchange System (IGES)
Graphics Kernel System
Graphics Kernel Standard (GKS)
Virtual Device Metafile (VDM)
Virtual Device Interface (VDI)
Ada Device Drivers
NAPLPS Device Drivers
North American Presentation Level Protocol Standard (NAPLPS)
Standard Device
Non-Standard Device
NAPLPS Device

Figure 4.3 GRAPHICS EXAMPLE OF FUNCTIONAL INTERFACE STANDARDS

4.4.1.3 REPOSITORY

STARS will maintain a high quality on-line, trusted, mandatory-access-controlled software repository with continuously improving human interface capabilities. The repository will be used to make available processes as well as the tools and tool collections supporting them. Software processes supporting the creation and evolution of software systems can be used equally well to support the development and maintenance of automated environments or end-application systems. General approaches and specific methods will be included in the repository so they can be accessed by the STARS community.

Special attention will be given to including processes supporting reusability. In particular, the repository will hold processes, tools and tools collections needed to make use of the repository itself. The repository will provide access to a variety of reusable software system fragments. Assistance will be needed to identify the fragments pertinent to a particular project, evaluations, adapt the chosen fragment as necessary, and integrate them with other (new or reused) fragments. The repository will, therefore, contain and provide access to the selection and assembly processes supporting the use of elements in the repository for the development of automated environments and end-application systems.

Initially STARS will use the Ada repository on SIMTEL20. As soon as possible, provision will be made to establish and support a STARS repository accessible through MILNET.

4.4.1.4 DESIGN BY SUCCESSIVE REFINEMENT

The package feature of the Ada language supports the separation of the design process from the coding process. The application of tools that automatically create package body stubs from package specifications allow the Ada compiler to be used as a design tool for checking design completeness and consistency. The capability introduces the opportunity for a process called design by successive refinement in which compilable designs can be ever extended with additional capabilities while simultaneously providing a continuing check on completeness and consistency.

These capabilities need to be part of the standard design process. Additionally tools need to be developed to support the process beginning with existing code as well as requirements statements.

4.4.2 RELIABLE SOFTWARE TECHNOLOGY

Reliability will be founded in computer aided methods, Ada-technology domain extensions to include a systems view, formal mehods for functional correctness (and security), and automated configuration control processes for a new reusable-software evolution concept.

4.4.2.1 COMPUTER AIDED METHODS

A number of tools are available in the Ada repository on SIMTEL20 to aid in the software development process. Another class of tools that could contribute to more reliable software in the near-term are known as validation capabilities as represented by the Stanford University, ANNA (ANNotated Ada) specification and validation system. Capabilities of this software need to be identified and integrated into the system environment.

4.4.2.2 DOMAIN INTEGRATION

The classical software engineering process has been generally defined in terms of a series of activities proceeding sequentially from requirements, to specifications, to design to coding to testing, and finally to maintenance. The Ada language has tightly coupled the design and coding steps through the Ada package feature. A number of compiler builders use the DIANA (Descriptive Intermediate Attribute Notation for Ada) internal representation of the Ada language and a technology called directed attribute trees to describe relationships. The directed attribute tree technology, now supported by the Ada compiler industry base consisting of more than 40 companies, includes methods similar to those used by the university and academic research community to relate policy statements, to policy models, to top level specifications and detailed design specifications with formal verification techniques. This observation has prompted the proposal to explore the integration of Ada package designs, upward to the top level specification and requirements domains by involving relationship at the internal representation level. The new industrial base would be used in developing these concepts.

There are a number of fundamental new research issues associated with this concept, but a demonstration of feasibility could open a powerful new approach to integrating the specification and design process that would appear to make a major contribution to software reliability.

The extent to which Ada has been useful in the STARS/VHSIC sponsored effort to describe hardware design, as a first step toward an integrated system design process addressing both software and hardware, suggests that the design process could be extended downward to include hardware and systems domains in an integrated way. Such a downward unification would also contribute to system reliability. These sorts of opportunities will be explored in STARS.

4.4.2.3 FORMAL METHODS

Formal mathematical methods have been used to "verify" that two different descriptions (e.g., top level specification and detailed design) of the same system are completely consistent. Formal methods offer one approach to improve software reliability. Their use is prominent in the development of trusted multi-level secure operating systems. The methods have also been used for correctness and reliability analysis of some critical very high value MCCR systems. Today's methods are, however, not affordable for widespread application. As one progresses from the functional requirement, to top level specification, to detail design, to code, and to execution, the process of "verifying" full and exact correspondence between successive views of the same system becomes increasingly difficult. The research needed to develop and productize formal methods for widespread application would exceed the funding levels currently in STARS. STARS will maintain close coordination with those DoD Agencies with primary responsibilities acquiring such capabilities. Some STARS projects in these areas are planned within available funding limitations. At a minimum the Ada Joint Program Office effort to develop a formal semantics for Ada must continue with U.S. participation. STARS may fund some development of Boyer-Moore theorem provers in Ada and for Ada. These activities will be coordinated with those of the National Computer Security Center and organizations using formal methods for functional correctness analyses.

4.5 RISK REDUCTION

Experience in shadow demonstrations, environment applications and technology development will provide the basis for several briefings for industry annually at which fundamental research issues will be identified for creative industry research. Pre-prototype projects will provide a new model for dealing with high-risk MCCR acquisition issues. The present intensive acquisition approaches have historically focused on the development of elaborate specifications before the fundamental problems are identified and understood. This leads to partially or completely ineffective software and systems or runaway development costs.

SECTION 5 - TECHNICAL GUIDANCE

The following general technical guidance will apply to all STARS activities:

• The STARS program will place strong emphasis on the development of compilable specifications, designs and executable code in Ada. To the maximum extent feasible, all code will be in machine independent Ada. The syntax and semantics of Ada will be used to the maximum in all supporting and descriptive work. Design will be developed using exact Ada as the Program Design Language. The designs should be compilable using any validated Ada compiler. Design information should be contained in compilable and executable Ada rather than textual comments. No comment information will be included in code products which are duplicated in processable Ada.

• All deliverable code will be compiled and executed by Ada compilers operating on at least two different instruction set architectures to encourage developers to include machine independence considerations from the design forward.

• Compliance with DoD Directive 5000.31 or equivalent policy statements on the use of Ada is mandatory unless justified and explicitly approved in writing. Failure to comply will result in the decrement of a participating executing agency's following year funding by the dollar amount of projects not in compliance.

• DoD, national and international standards will be used and supported. Emerging standards will be monitored and incorporated when possible.

• Ada will be the "command language" for interfacing between programs and for interfacing to the menus, and displays at the man-machine interface.

• Design should proceed by refinement and modification of product code. The design statements should be suitable for maintenance throughout the software life so that the final code contains the accurate design information.

• The description of systems to be verified will be in an Ada form. Such restrictions and modifications to the exact language as are required for the particular technique under investigation will be kept to a minimum and explicitly justified.

• Maximum use will be made of automated tools. Because of the language commonality (i.e., all tools and environments will be in and for Ada) the construction and sharing of tools will be facilitated.

• As a general practice, studies and activities that do not lead to, or accompany, the delivery of executable products in and for the Ada technology, shall not be funded.

• All proposals for contract or government internal work will be judged on the technical merits of the writeup that should include, but not be limited to:

- Statement of problem or opportunity
- Identification of prior and related work
- Identification of potential applications and users
- Summary of pertinent literature
- Summary of applicable standards
- Nature of expected contributions or benefits
- Basis for expected contributions
- Cost estimates
- Basis for cost estimates
- Demonstration that work has not been done

SECTION 6 - PRODUCTS AND MILESTONES

The STARS program will lead to the delivery of a number of products. These will include:

6.1 SHADOW MCCR SYSTEMS

At least 12 MCCR Shadow demonstrations will be conducted at a rate of 3 per year beginning in FY 1987. The demonstration will result in delivery of operational, mission quality Ada software with a goal that the products can be used by the Services.

6.2 SOFTWARE DEVELOPMENT ENVIRONMENTS

Several prototype software engineering development and maintenance environments, each specialized for a particular applications domain, will be delivered in FY 1988. These environments will be delivered as fully operational systems for the Services in FY 1991.

A sizable family of software engineering tools to support a reliable and adaptable software technology consistent with an evolutionary software-first approach to systems acquisition will be delivered and refined during the STARS program.

6.3 TECHNOLOGY DEVELOPMENT

In the summer of 1984, the WIS Joint Program Office contracted through Naval Ocean System Center (NOSC) for the development of Ada software. Proposals were sought from industry against 57 categories of software including tools and applications. The work has lead to the delivery of software, some of which is listed in Section 4.4.1.1 above.

STARS will build on the successful NOSC initiative, and extend the Ada tools currently available at the Ada repository on SIMTEL20, by developing a significant quantity of reusable foundation software capabilities in Ada during FY 1986 and FY 1987. Many of the common capabilities will have direct application to the BLOCK C WWMCCS Information System Joint Program, as well as the STARS environment and MCCR Shadow application developments.

STARS will seek to unify, using Ada, a meaningful set (e.g., those covering operating systems, data base, graphics, network, documentation, etc.) of functional interface standards as a means of establishing a reusable software technology. This effort will continue throughout the STARS program but will be sufficiently mature by FY 1988 to begin adapting the foundation capabilities to these interface standards.

STARS will develop and use improved repository capabilities accessible over MILNET to store and make available the STARSproducts and technology. In FY 1986, the Ada repository at SIMTEL20 will be used. A STARS repository will be brought on-line in FY 1987 and maintained through the STARS program.

A special effort will lead to tools and processes to support an adaptable and reliable software engineering approach. The specific tools will be better defined as the program matures.

6.4 FUNDAMENTAL RESEARCH

STARS will seek to pioneer and demonstrate a new approach for dealing with high risk software issues in MCCR systems acquisition. The approach will result in several briefings for industry each year on fundamental issues needing resolution. Additionally a rapid response mechanism will be established to fund creative solution proposals from industry. STARS will identify high-risk areas requiring creative, new solutions. Rather than attempting to write detailed specifications as the outcome of the systems engineering process, STARS will instead use pre-prototype and prototype activities leading to executable, feasibility demonstrations of solution approaches. These demonstrations will provide the basis for credible specifications in support of process and product development, that in turn support mission applications or environment integration.

6.5 DETAILED MILESTONES

More detailed contract milestones will be provided by the STARS Joint Program Office with the competing-primes lead contractor acquisition documents. Also the Service implementation plans will have more detailed milestones for the Service portion of the research and development program.

SECTION 7 - REFERENCES

1. CHARTER, Software Technology for Adaptable, Reliable Systems (STARS), Office of the Under Secretary of Defense, R. D. DeLauer, 1 November 1984.

2. Memorandum for Joint Logistics Commanders, "Staffing and Management of DoD Software Programs," Deputy Secretary of Defense, W. H. Taft, IV; 12 August 1985.

3. STARS Program Requirements Document, Prepared by Joint Logistics Commanders - Computer Resources Management, Recommended to Deputy Secretary of Defense, January 1986.

4. STARS Program Management Plan, Submitted by STARS Director, STARS Joint Program Office, May 1986.

5. "DoD Computing Activities and Programs - 1985 Specific Market Study," Market Planning Reference Publications, The Requirements Committee, Government Division, Electronic Industries Association (EIA), December 1985. This study updates a popular 1980 EIA 10 year study of DoD Digital Data Processing. The numbers used in Section 1.1 represent both EIA studies. DoD's annual expenditures for MCCR are projected to increase from $2.82B (1980 EIA study) in 1980 to $11.17B (1980 EIA estimate) - $11.41B (1985 EIA estimate) in 1985 and then on up to $32.10B (1980 EIA projection) - $25.59B (1985 EIA projections).

6. US Army Science Advisory Board (SAB), 1983 Summer Study on Acquiring Army Software, December 1983, Department of the Army, Assistant Secretary of the Army, Research, Development, and Acquisition, Washington, D.C. 20310.

7. USAF Science Advisory Board Ad Hoc Committee on the High Cost and Risk of Mission Critical Software, December 1983.

8. Defense Science Board (DSB) Task Force on Software, Chartered by the USDRE, 2 November 1984. (The Task Force Chairman, Dr. Fred Brooks, presented a preview of DSB finding to DUSD (R&AT) in April 1986).

9. "An Assessment of the STARS Program September-October 1985," prepared for the USDR&E by the Institute for Defense Analyses, in three volumes, December 1985.

10. Greene, J. S. Jr., "The National Computer Security Center", SIGNAL, Journal of the Armed Forces Communications and Electronic Association, September 1985.

11. Schill, J., Smeaton, R., and Jackman, R., The Conversion of Command & Control Software to Ada: Experiences

and Lessons Learned, Ada Letters, SigAda Special Interest ACM, Vol IV, Issue 4, January-February 1985.

12. Whitaker, W.A., Keynote Address, Second International Ada Applications and Environments Conference, sponsored by the Institute of Electrical and Electronics Engineers, Miami, Florida, 8-10 April 1986.

13. Boehm, B., Software Engineering Economics, Prentice-Hall, NY, 1981.

14. Boehm, B. and Standish, "Software Technology in the 1990's Using an Evolutionary Paradigm", IEEE COMPUTER, Vol 16, No. 11, pp 30-38, November 1983.

15. Reference Manual for the Ada Language, ANSI/MIL-STD-1815A, Department of Defense, February 17, 1983.

16. Defense System Software Development Military Standard, DoD-STD-2167, Department of Defense, 4 June 1985.

17. Nash, S. H., Redwine, S. T, Jr., "Information Interface Related Standards, Guidelines, and Recommended Practices", IDA Paper P-1842, Institute for Defense Analyses, 1801 N. Beauregard St., Alexandria, VA 22311, July 1985.

18. Strategy for the Software Initiative (STARS), DUSD (R&AT)/Dir CSS, March 1983.

19. "Command Control Information Processing for the 1980s", CCIP-85, Published in 12 volumes, United States Air Force, September 1972.

APPENDIX A

FOUNDATION CAPABILITIES

Foundation capabilities to be developed in FY 1986 and 1987 will be applicable to the preparation of both automated environments and end-applications software systems. General capabilities are described in Section A.1. below. Capabilities specifically useful for automated environments and directed toward design support, are discussed in Section A.2. Foundations for end-application systems are discussed in A.3.

A.1 General Fragments

The general components, applicable to the preparation of both automated environments and end-application systems, fall into six categories:

-- Command languages.
-- Text processing.
-- Database.
-- Operating systems.
-- Graphics.
-- Network protocols.

These are discussed in the following subsections.

A command language provides a communication interface between a computer system and its users. It must enable each user to solve problems on a semantic level appropriate to those problems. In most DoD software developments, there are usually DoD developer/users of varying levels of expertise who must communicate with the host development system and with each other. Their interactions need to be facilitated by a common, uniform, consistent interface.

Activities in this category will provide prototypes of a variety of command languages and command language support facilities. The prototypes should provide multiple options for program control and allow the user to manipulate the system, control the session and data, manage processes and resources, and package primitive commands. These prototypes will allow for the development of uniform, consistent man/machine interfaces. The separable functions of the command language capabilities should be partitioned into packages to facilitate maintenance.

Specific activities falling within this area could include:

-- Emulation of a form-terminal on a page-terminal.
-- Command string parser/interpreters.
-- Command language implementations conforming to existing standards.

-- Form-based implementations of standard command languages.
-- Form and menu generation tools for bitmap-, page- and forms-terminals.
-- Guidelines for interface design.
-- Interface to existing components.

A.1.2 Document/Text Preparation

The objectives for these activities should be to develop a prototype Ada-based "document management" system with capabilities for, but not limited to, word processing, providing output with multiple-type fonts, user-oriented "help" messages tailored to the operations being performed, and user history and expertise. Design issues to be resolved should include the provision for compatibility with multiple subsystems that may or may not have been completely defined/developed, the ability to use a variety of input/output devices, the capability of using textual syntax/semantics to provide assistance in document preparation.

Prototypes here should include those for demonstration of (1) automatic generation of text-based systems, (2) user interfaces for text processing, and (3) integrated packages of writer's workbench tools. Emphasis should be towards "what you see is what you get (WYSIWYG)" systems which permit multiple views. Preparation of electronic documents and of reasonable quality paper documents should be prototyped first. All of these prototypes should be extensible. These systems should assume and plan for advanced input/output technologies, should be input/output device independent, and should use virtual input/output devices.

Specific projects falling within this area could include:

-- Text/editor formatter generators.
-- Interface modules supporting common document preparation tasks.

A.1.3 Database Support

The objectives for these activities should include prototype demonstration of potentially standard Ada interface for a portable commercial off-the-shelf DBMS. The prototype Ada-DBMS might provide capabilities for data base definition, including the provision of multiple views (i.e., relational, network hierarchical) of the same database; efficient retrieval and update of individual objects or groups of logically related database objects; authorization control to the field or attribute level, capable of expansion to provide multi-level security control (e.g., prohibiting a query that is authorized to access aggregated data at a given classification from generating secondary accesses to data objects at a higher security classification); multiple interfaces, including programming and query languages, screen-oriented command language/displays, bulk, load/unload facilities, report writer and application generator; multiple user access with backup and recover; and database administration and control.

Design issues which might be resolved here include support of fully distributed access, including partial or full replication of objects; the ability to incorporate "database machines" or "back-end database processor" technology for retrieval operations; expert system interfaces; verifiable security protection including the ability to operate in a multi-level security environment; efficient processing of multiple views or schema; and hardware independence and portability to include optimization of retrieval strategies.

Thirteen potential database management system tools/components to aid in the creation and tailoring of support for a range of high-level systems should be considered for prototyping. These include:

-- An indexed file access package.
-- A concurrency control package.
-- A work station relational database system package.
-- A multi-user relational database system package.
-- A relational database design tool kit.
-- A view definition facility.
-- A view query and update facility.
-- An authorization package.
-- A database query optimizer/compiler.
-- A distributed database access package.
-- Distributed database protocol management.
-- Database operational tools, and
-- Database application tools.

Based on a set of criteria which emphasize independence, general understanding of functionality and implementation methods, and immediate usefulness, three components can be identified as candidates for early implementation: indexed file access package, work station relational database system, and database application tools. It appears reasonable to prototype these components in parallel.

A.1.4 Operating System Fragments

A set of operating system component prototypes could be both reliable and portable if written in Ada. Such components would include discretionary and mandatory access control at the B3 level, and account for uniprocessors, multi-processors, and multi-computer systems. These prototypes could initially support one local area network. However, their design and implementation might not preclude, nor make more difficult their interaction with other Local Area Networks (LAN's). A prototype set of operating system components could have the following attributes: fault tolerance, survivability, multi-level security at or above the B3 level, portability of applications and system software across a wide variety of machine sizes and types, single and multi-thread machines, multiple priorities, real-time processing, and fully integrated databases.

Operating system prototyping and developmental activities could concern a variety of topics, including:

-- Kernel operating systems.
-- Program execution support module.
-- File manager.
-- Authentication server.
-- Time synchronization agent.
-- Transaction manager.
-- Inter-process communication support.
-- Alias processes supporting access to remote LAN's.
-- Print server.
-- Input/output drivers.
-- Multi-window system.
-- Logging and auditing facilities.

A.1.5 Graphics Support

A high-level conceptual model of a graphics support system could be prototyped. The model could be a pipeline showing the flow of information and control between one or more application programs and the graphics display and input hardware. The model could illustrate specific components of a graphics support system. Primarily, these components would be the visual objects (their definition and manipulation), a window management system, and an image generation system.

An object-oriented approach could be adopted in which the application program deals with high-level "visual objects," such as menus and icons, rather than just primitives as points and lines. Possible activities for prototyping specific parts of this model include:

-- Visual object specification techniques.
-- Interfaces supporting reuse of visual objects.
-- Coordination/management support for descriptions employing visual objects.
-- Image generation capabilities.
-- Window managers.
-- Computer graphics interfaces.
-- Graphics command, input, output processors.

A.1.6 Network Support

Transfer of a variety of types of information to both local and remote users and systems is a common characteristic of many DoD development environments and end-application systems. Access to these systems' capabilities, both the functions and data, are likely to be accomplished using a LAN architecture in both automated environment and end-application systems. Access to other sites and remote systems and data are likely to be accomplished using intercomputer networking capabilities through the Defense Data Network (DDN).

The needs for network services necessitate tools to build integrated communications subsystems. To realize the benefits of the distributed processing concept, any number of components (hardware and software) from different sources must be able to communicate among themselves through the use of predetermined protocols. Current DoD protocols are likely to be used in the near term. However, activity in the international standardization community indicates that at some time in the future, ISO protocols may be used for network communication functions. As services are expanded the need for new applications and lower-level protocols grows. Given the experience of the ARPANET, methods which can reduce the complexity and resources needed for new protocol specification and implementation are likely to be necessary for developing and using many DoD systems. Use of the DDN as the long-haul backbone raises the questions of how inter-communications routing will be accomplished when additional factors beyond shortest distance must be accommodated. Prototyping activities in this area could include:

-- Ada-based implementation of ISO and DoD transport and internet protocols.
-- Generators of Ada protocol software.
-- Multi-variable objective functions supporting the optimization of network routing.

A.2 Automated Environment Fragments

Some of the fragments in the repository will only serve the need to develop automated environments. Particularly important in this regard are fragments supporting preliminary and detailed design.

A.2.1 Design Description and Analysis

Careful attention to design is essential to achieving the benefits of software reliability, efficiency, maintainability, and portability. Automated support can substantially improve the design process and the fragments developed in this set of activities will develop currently promising concepts for providing this automated support.

The emphasis in the near term is upon design. This allows early results by capitalizing on technology that has relatively apparent value but has not been brought to the Ada arena. It also provides foundation for later work on tools supporting other life cycle phases.

Example activities in this area are:

-- Advanced Ada Program Design Languages (PDL's).
-- PDL support tools.
-- Graphics support tools for software design.

END

12-86

DTIC